Compositional design methodology with constraint Markov chains

Benoit Caillaud, Benoît Delahaye, Kim Guldstrand Larsen, Axel Legay, Mikkel L. Pedersen, Andrzej Wasowski

Compositional Design Methodology with Constraint Markov Chains

Benoît Caillaud*, Benoît Delahaye†, Kim G. Larsen‡, Axel Legay*, Mikkel L. Pedersen‡ and Andrzej Wąsowski§

*INRIA/IRISA
Rennes, France
{benoit.caillaud,axel.legay}@irisa.fr

†Université de Rennes 1 / IRISA
Rennes, France
benoit.delahaye@irisa.fr

‡Aalborg University
Denmark
{kgl,mikkelp}@cs.aau.dk

§IT University
Copenhagen, Denmark
wasowski@itu.dk

***Abstract*— Notions of specification, implementation, satisfaction, and refinement, together with operators supporting stepwise design, constitute a specification theory. We construct such a theory for Markov Chains (MCs) employing a new abstraction of a Constraint MC. Constraint MCs permit rich constraints on probability distributions and thus generalize prior abstractions such as Interval MCs. Linear (polynomial) constraints suffice for closure under conjunction (respectively parallel composition). This is the first specification theory for MCs with such closure properties. We discuss its relation to simpler operators for known languages such as probabilistic process algebra. Despite the generality, all operators and relations are computable.**

I. Introduction

Modern systems are big and complex, resulting from assembling multiple components. The components are designed by teams, working independently but with a common agreement on what the interface of each component should be. As a consequence, mathematical foundations that allow to reason at the abstract level of interfaces in order to infer global properties are an active research area known as *compositional design* [1]. Within this area specification theories provide a modeling language that allows designing, evolving and advisedly reusing components with formal guarantees. In a logical interpretation, interfaces are specifications and systems/components that implement a specification are models/implementations. There is an agreement that a good theory should support the following requirements:

1) *Consistency and Satisfaction.* It should be decidable whether a specification admits at least one implementation, and whether a system implements a specification.
2) *Refinement. Refinement* of specification expresses inclusion of sets of implementations, and therefore allows to compare richness and precision of specifications.
3) *Structural composition.* A theory should provide a combination operator on specifications, reflecting the standard composition of systems by, *e.g.* parallel product.
4) *Logical composition/conjunction.* Different aspects of systems are often specified by different teams. The issue of dealing with multiple aspects of multiple viewpoints is thus essential. It should be possible to represent several specfications (viewpoints) for the same system, and to combine them in a logical/conjunctive fashion.
5) *Incremental Design.* A theory should allow incremental design (composing/conjoining specifications in any order) and independent implementability (composable specifications can always be refined separately) [2].

For functional analysis of discrete-time non-probabilistic systems, the theory of Modal Transition Systems (MTS) [3] provides a specification formalism supporting refinement as well as conjunction and parallel composition. It has been recently applied to construct interface theories [4], [5], which are extensions of classical interface automata proposed by de Alfaro et al. [6], [7], [8].

As soon as systems include randomized algorithms, probabilistic protocols, or interact with physical environment, probabilistic models are required to reason about them. This is exacerbated by requirements for fault tolerance, when systems need to be analyzed quantitatively for the amount of failure they can tolerate, or for the delays that may appear. As Henzinger and Sifakis [1] point out, introducing probabilities into design theories allows assessing dependability of IT systems in the same manner as commonly practiced in other engineering disciplines.

Generalizing the notion of MTSs to the non-functional analysis of probabilistic systems, the formalism of Interval Markov Chains (IMCs) was introduced [9]; with notions of satisfaction and refinement generalizing probabilistic bisimulation. Informally, IMCs extend Markov Chains by labeling transitions with *intervals* of allowed probabilities rather than concrete probability values. Implementations of IMCs are Markov Chains (MCs) whose probability distributions match the constraints induced by the intervals. IMCs is known to be an efficient model on which refinement and composition can be performed with efficient algorithms from linear algebra. Unfortunately, as we shall now see, the expressive power of IMCs is inadequate to support both logical and structural composition.

Consider two IMCs, S_1 and S_2, in Figure 1 specifying different probability constraints related to the height H and weight W of a given person. Attempting to express the conjunction $S_1 \wedge S_2$ as an IMC by a simple intersection of bounds gives $z_1 \leq \frac{1}{2}$, $\frac{1}{6} \leq z_2 \leq \frac{1}{2}$, $\frac{1}{8} \leq z_3$ and $\frac{1}{6} \leq z_4$. However, this naive construction is too coarse: whereas $(z_1, z_2, z_3, z_4) =$

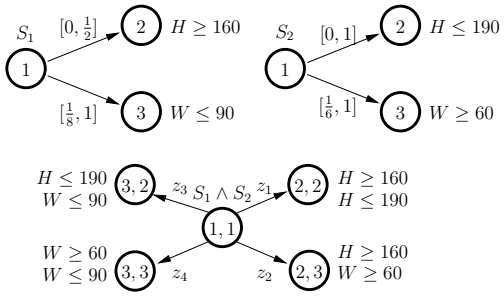


Fig. 1: IMCs showing non-closure under conjunction

$(\frac{1}{2}, \frac{1}{6}, \frac{1}{8}, \frac{5}{24})$ satisfies the constraints the resulting overall probability of reaching a state satisfying $H \geq 160$, i.e. $z_1+z_2 = \frac{2}{3}$, violates the upper bound $\frac{1}{2}$ specified in S_1. What is needed is the ability to express dependencies between the probabilities z_1, z_2, z_3, z_4 besides that of being a probability distribution $(z_1+z_2+z_3+z_4 = 1)$. The correct conjunctive combination is expressed by three following constraints, exceeding the expressive power of IMCs: $z_1+z_2 \leq \frac{1}{2}$, $\frac{1}{8} \leq z_3+z_4$, $\frac{1}{6} \leq z_2+z_4$. A similar example shows that IMCs are also not closed under parallel composition, either.

One way to approach this problem could be to work with two types of specifications: IMCs for refinement and structural composition, and a probabilistic logic such as PCTL [10] on which a logical conjunction is naturally defined. Such a solution is clearly not satisfactory. Indeed, it is not clear how one can synthesize a MC (an implementation) that satisfies two PCTL formulas. It is also not possible to structurally compose two logical PCTL formulas.

The solution promoted in this paper is to enrich the model of IMCs. More precisely, we introduce *Constraint Markov Chains* (CMCs) as a foundation for component-based design of probabilistic systems. CMCs are a further extension of IMCs allowing rich constraints on the next-state probabilities from any state. Whereas linear constraints suffice for closure under conjunction, polynomial constraints are necessary for closure under parallel composition. We provide constructs for refinement, consistency checking, logical *and* structural composition of CMC specifications – all indispensable ingredients of a compositional design methodology.

In CMCs, each state is also labelled with a set of subsets of atomic propositions. Those propositions represent properties that should be satisfied by the implementation. The idea being that the satisfaction relation ensures that an implementation matches at least one of the subsets. This allows the specification to make additional assumptions on the behaviors of the implementation. Hence, at the level of specification, our model thus presents choices on subsets of actions. However these choices are independent from the probabilistic ones in the sense that any CMC whose states are labelled with a set of subsets of atomic propositions can be turned to an equivalent (in terms of set of implementations) CMC whose states are labeled with a single subset of atomic propositions. There, choices between the subsets of actions disappear. It is thus not surprising that our notion of parallel composition is following the widely accepted *principle of separation of concerns*. The idea is to separate parallel composition of probability distributions from synchronization on sets of actions. This separation can be found in probabilistic specification theories that have probabilistic automata as an underlying semantic model [11], [12], [13], [14]. In fact, we show how probabilistic automata can be represented as CMCs, and how the traditional notions of parallel composition on such model can be derived in our framework with precongruence properties obtained for free. This latter result shows that CMCs capture computational structure of known models and operators, laying down a basis for studying shared properties of many probabilistic automata based languages. As already mentioned, we exemplify this by showing how precongruence properties for composition of probabilistic automata and known refinements can be obtained by reductions to CMCs.

The notions of satisfaction and strong/weak refinements for CMCs conservatively extend similar notions for IMCs [15], [9]. We characterize these relations in terms of implementation set inclusion. In particular, in the main theorem, we prove that for deterministic CMCs weak and strong refinements are complete with respect to implementation set inclusion. In addition, we provide a construction, which for any CMC S returns a deterministic CMC $\varrho(S)$ containing the models of S. Refinement relations are not complete for non-deterministic CMCs, but one can show that the weak refinement is more likely to coincide with implementation set inclusion in such a context. We show that refinement between CMCs with polynomial constraints can be decided in essentially single exponential time.

Structure of the paper. In Section II, we introduce the concept of CMCs and a satisfaction relation with respect to Markov Chains. Consistency, refinement and conjunction are discussed in Section III. Structural composition is introduced in Section IV. In Section V, we introduce deterministic CMCs and show that, for this class of CMCs, strong and weak refinements coincide with inclusion of implementation sets. Section VI discusses the class of polynomial CMCs, which is the smallest class of CMCs closed under all the compositional design operations. Section VIII concludes the paper with related and future work. Due to space constraints, some algorithms and proofs are given in a long version of this paper [16].

II. Constraint Markov Chains

Let A, B be sets of propositions with $A \subseteq B$. The *restriction of* $W \subseteq B$ *to* A is given by $W{\downarrow_A} \equiv W \cap A$. If $T \subseteq 2^B$, then $T{\downarrow_A} \equiv \{W{\downarrow_A} \mid W \in T\}$. For $W \subseteq A$ define the *extension of* W *to* B as $W{\uparrow^B} \equiv \{V \subseteq B \mid V{\downarrow_A} = W\}$, so the set of sets whose restriction to A is W. Lift it to sets of sets as follows: if $T \subseteq 2^A$ then $T{\uparrow^B} \equiv \{W \subseteq B \mid W{\downarrow_A} \in T\}$. Let $M, \Delta \in [0,1]^{n \times k}$ be two matrices and $x \in [0,1]^{1 \times k}$ be a vector. We write M_{ij} for the cell in ith row and jth column of M, M_p for the pth row of M, and x_i for the ith element of x.

Finally, Δ is a *correspondence matrix* iff $0 \leq \sum_{j=1}^{k} \Delta_{ij} \leq 1$ for all $1 \leq i \leq n$.

Definition 1 (Markov Chain)**.** $P = \langle \{1, \ldots, n\}, o, M, A, V \rangle$ *is a* Markov Chain *if* $\{1, \ldots, n\}$ *is a set of states containing the initial state* o*,* A *is a set of atomic propositions,* $V : \{1, \ldots, n\} \to 2^A$ *is a state valuation, and* $M \in [0,1]^{n \times n}$ *is a probability transition matrix:* $\sum_{j=1}^{n} M_{ij} = 1$ *for* $i = 1, \ldots, n$*.*

We now introduce *Constraint Markov Chains* (CMCs for short), a finite representation for a possibly infinite set of MCs. Roughly speaking, CMCs generalize MCs in that, instead of specifying a concrete transition matrix, they only constrain probability values in the matrix. Constraints are modelled using a *characteristic function*, which for a given source state and a distribution of probabilities of leaving the state evaluates to 1 iff the distribution is permitted by the specification. Similarly, instead of a concrete valuation function for each state, a *constraint on valuations* is used. Here, a valuation is permitted iff it is contained in the set of admissible valuations of the specification.

Definition 2 (Constraint Markov Chain)**.** *A* Constraint Markov Chain *is a tuple* $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$*, where* $\{1, \ldots, k\}$ *is a set of states containing the initial state* o*,* A *is a set of atomic propositions,* $V : \{1, \ldots, k\} \to 2^{2^A}$ *is a set of admissible state valuations and* $\varphi : \{1, \ldots, k\} \to [0,1]^k \to \{0,1\}$ *is a* constraint function *such that if* $\varphi(j)(x) = 1$ *then the* x *vector is a probability distribution:* $x \in [0,1]^k$ *and* $\sum_{i=1}^{k} x_i = 1$*.*

An *Interval Markov Chain* (IMC for short) [9] is a CMC whose constraint functions are represented by intervals, so for all $1 \leq i \leq k$ there exist constants α_i, β_i such that $\varphi(j)(x) = 1$ iff $\forall 1 \leq i \leq k,\ x_i \in [\alpha_i, \beta_i]$.

Example. *Two parties, a customer and a vendor, are discussing a design of a relay for an optical telecommunication network. The relay is designed to amplify an optic signal transmitted over a long distance over an optic fiber. The relay should have several modes of operation, modelled by four dynamically changing properties and specified by atomic propositions* a*,* b*,* c*, and* e*:*

Atomic propositions in the optic relay specifications		
a	ber $\leq 10^{-9}$	bit error rate lower than 1 per billion bits transmitted
b	br > 10Gbits/s	The bit rate is higher than 10 Gbits/s.
c	$P < 10$W	Power consumption is less than 10 W.
e	Standby	The relay is not transmitting.

The customer presents CMC S_1 *(Figure 2a) specifying the admissible behaviour of the relay from their point of view. States are labelled with formulas characterizing sets of valuations. For instance, "*$(a + b + c \geq 2) \wedge (e = 0)$*" at state* 2 *of* S_1 *represents* $V_1(2) = \{\{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$*, where* a*,* b*,* c*, and* e *range over Booleans. State 1 specifies a standby mode, where no signal is emitted and only marginal power is consumed. State 2 is the high power mode, offering a high signal/noise ratio, and hence a high bit-rate and low error rate, at the expense of a high power consumption. State 3 is the low power mode, with a low power consumption, low bit-rate and high error rate. The customer prescribes that the probability of the high power mode (state 2) is higher than* 0.7*. The vendor replies with CMC* S_2 *(Figure 2b), which represents possible relays that they can build. Because of thermal limitations, the low power mode has a probability higher than* 0.2*.*

A state u of S is (directly) *reachable* from a state i if there exists a probability distribution $x \in [0,1]^k$ with a nonzero probability x_u, which satisfies $\varphi(i)(x)$.

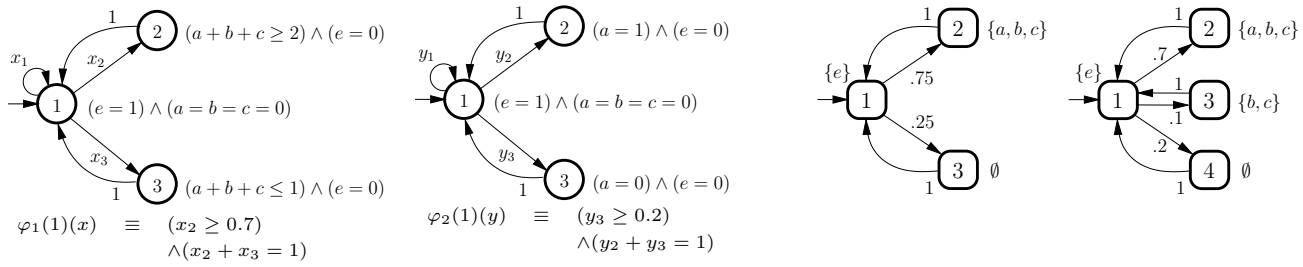
We relate CMC specifications to MCs implementing them, by extending the definition of satisfaction presented in [9] to observe the valuation constraints and the full-fledged constraint functions. Crucially, like [9], we abstract from syntactic structure of transitions—a single transition in the implementation MC can contribute to satisfaction of more than one transition in the specification, by distributing its probability mass against several transitions. Similarly many MC transitions can contribute to satisfaction of just one specification transition.

Definition 3 (Satisfaction Relation)**.** *Let* $P = \langle \{1, \ldots, n\}, o_P, M, A_P, V_P \rangle$ *be a MC and* $S = \langle \{1, \ldots, k\}, o_S, \varphi, A_S, V_S \rangle$ *be a CMC with* $A_S \subseteq A_P$*. Then* $\mathcal{R} \subseteq \{1, \ldots, n\} \times \{1, \ldots, k\}$ *is a* satisfaction relation *between states of* P *and* S *iff whenever* $p \mathcal{R} u$ *then*

1) $V_P(p)\downarrow_{A_S} \in V_S(u)$*, and*
2) *there exists a correspondence matrix* $\Delta \in [0,1]^{n \times k}$ *such that*
   - *for all* $1 \leq p' \leq n$ *with* $M_{pp'} \neq 0$*,* $\sum_{j=1}^{k} \Delta_{p'j} = 1$*;*
   - $\varphi(u)(M_p \times \Delta)$ *holds and*
   - *if* $\Delta_{p'u'} \neq 0$ *then* $p' \mathcal{R} u'$*.*

We write $P \models S$ iff there exists a satisfaction relation relating o_P and o_S, and call P an *implementation* of S. The set of all implementations of S is given by $[\![S]\!] \equiv \{P \mid P \models S\}$. Rows of Δ that correspond to reachable states of P always sum up to 1. This is to guarantee that the entire probability mass of implementation transitions is allocated. For unreachable states, we leave the corresponding rows in Δ unconstrained. P may have a richer alphabet than S, in order to facilitate abstract modelling: this way an implementation can maintain local information using internal variables. Algorithms to decide satisfaction are particular cases of algorithms to decide *refinement* between CMCs. See the next section.

Example. *We illustrate the concept of correspondence matrix between Specification* S_1 *(given in Figure 2a) and Implementation* P_2 *(given in Figure 2d). The CMC* S_1 *has three outgoing transitions from state 1 but, due to constraint function in* 1*, the transition labelled with* x_1 *cannot be taken (the constraint implies* $x_1 = 0$*). The probability mass going from state 1 to states 2 and 3 in* P_2 *corresponds to the probability allowed by* S_1 *from its state* 1 *to its state* 2*; The redistribution is done with the help of the matrix* Δ *given in Figure 3c. The* i*th column in* Δ *describes how big fraction of each transition probability (for transitions leaving 1) is associated with probability* x_i *in* S_1*. Observe that the constraint function* $\varphi_1(1)(0, 0.8, 0.2) = \varphi_1(1)((0, 0.7, 0.1, 0.2) \times \Delta)$ *is satisfied.*



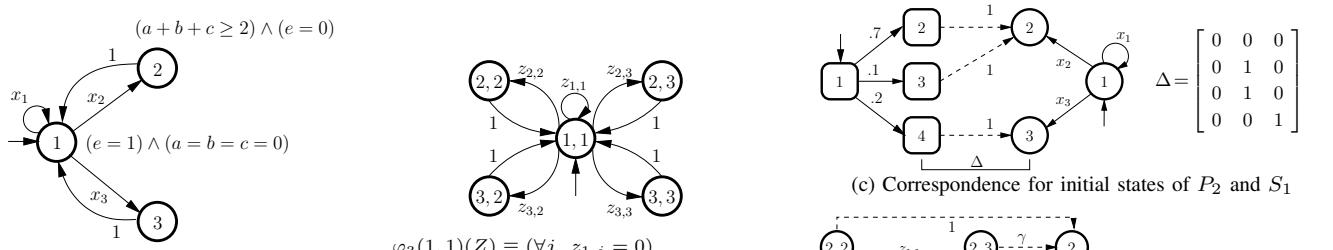
(a) CMC S_1, the customer specification of the optical relay

(b) CMC S_2, The manufacturer specification of the relay

(c) Markov Chain P_1 satisfying S_1 and S_2

(d) Markov Chain P_2 satisfying S_1 and S_2

Fig. 2: Two specifications (CMCs) and two implementations (MCs) of an optic relay



(a) CMC S_4 generalizing S_3, so $S_3 \preceq S_4$

(b) $S_3 = S_1 \wedge S_2$. Constraints on propositions, pairwise conjunctions of constraints of S_1 and S_2, are left out to avoid clutter

(c) Correspondence for initial states of P_2 and S_1

(d) Weak refinement for initial states of S_3 and S_4

Fig. 3: Examples of refinement, conjunction and satisfaction for CMCs

CMC semantics follows the Markov Decision Process (MDP) tradition [17], [18]. The MDP semantics is typically opposed to the Uncertain Markov Chain semantics, where the probability distribution from each state is fixed a priori.

States of CMCs are labeled with set of subsets of atomic propositions. A single set of propositions represents properties that should be satisfied by the implementation. A set of sets models a choice of properties, with the idea being that the satisfaction relation ensures that an implementation matches at least one of the subsets. This allows the specification to make additional assumptions on the behaviors of the implementation. For an implementation, in each state the discrete choice of proposition set and the probabilistic choice of successor are independent.

It turns out that any CMC whose states are labelled with a set of subsets of atomic propositions can be turned into an equivalent (in terms of sets of implementations) CMC whose states are labeled with sets that contains a single subset of atomic propositions. Hence working with sets of subsets of valutations is a kind of modeling sugar that can be removed with a transformation to the *single valuation normal form*.

Definition 4. *We say that a CMC is in a* Single Valuation Normal Form *if all its admissible valuation sets are singletons (* $|V(i)| = 1$ *for each* $1 \leq i \leq k$*).*

More precisely every consistent CMC with at most one admissible valuation in the initial state can be transformed into the normal form preserving its implementation set. A polynomial time normalization algorithm can be found in [16].

III. Consistency, Refinement and Conjunction

Consistency. A CMC S is *consistent* if it admits at least one implementation. We now discuss how to decide consistency. A state u of S is *valuation consistent* iff $V(u) \neq \emptyset$; it is *constraint consistent* iff there exists a probability distribution vector $x \in [0,1]^{1\times k}$ such that $\varphi(u)(x) = 1$. It is easy to see that if *each state* of S is both valuation and constraint consistent then S is also consistent. However, inconsistency of a state does not imply inconsistency of the specification. Indeed, an inconsistent state could be made unreachable by forcing the probabilities to reach it to zero. The operations presented later in this paper may introduce inconsistent states, leaving a question if a resulting CMC is consistent. In order to decide whether S is inconsistent, state inconsistencies are propagated throughout the entire state-space using a *pruning operator* β that removes inconsistent states from S. The result $\beta(S)$ is a new CMC, which may still contain some inconsistent states. The operator is applied iteratively, until a fixpoint is

reached. S is consistent if the resulting CMC $\beta^*(S)$ contains at least one state. The formal definition is given in [16].
It can be shown (see [16]) that pruning preserves the set of implementations.

Proposition 5. *Let S be a CMC. We have that $[\![S]\!] = [\![\beta(S)]\!]$.*

The fixpoint of β, and thus the entire consistency check, can be computed using a quadratic number of state consistency checks. The complexity of each check depends on the constraint language chosen.

Refinement. Comparing specifications is central to stepwise design methodologies. Systematic comparison enables simplification of specifications (abstraction) and adding details to specifications (elaboration). Usually specifications are compared using a *refinement* relation. Roughly, if S_1 refines S_2, then any model of S_1 is also a model of S_2.

We will now introduce two notions of refinement for CMCs that extend two well known refinements for IMCs [9], [15]. We not only generalize these refinements, but, unlike [9], [15], we also characterize them in terms of implementation set inclusion - also called *thorough refinement* - and computational complexity.

The strong refinement between IMCs, by Jonsson and Larsen [9], extends to CMCs in the following way:

Definition 6 (Strong Refinement)**.** *Let $S_1 = \langle\{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle\{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be CMCs with $A_2 \subseteq A_1$. A relation $\mathcal{R} \subseteq \{1,\dots,k_1\}\times\{1,\dots,k_2\}$ is a* strong refinement relation *between states of S_1 and S_2 iff whenever $v\,\mathcal{R}\,u$ then*

1) *$V_1(v)\downarrow_{A_2} \subseteq V_2(u)$, and*
2) *there exists a correspondence matrix $\Delta \in [0,1]^{k_1\times k_2}$ such that for all probability distribution vectors $x \in [0,1]^{1\times k_1}$ if $\varphi_1(v)(x)$ holds then*
   - *For all $1 \le i \le k_1$, $x_i \neq 0 \implies \sum_{j=1}^{k_2} \Delta_{ij} = 1$;*
   - *$\varphi_2(u)(x \times \Delta)$ holds and*
   - *if $\Delta_{v'u'} \neq 0$ then $v'\,\mathcal{R}\,u'$.*

We say that S_1 strongly refines S_2 iff $o_1\,\mathcal{R}\,o_2$.

Strong refinement imposes a "fixed-in-advance" correspondence matrix regardless of the probability distribution satisfying the constraint function. In contrast, the *weak refinement*, which generalizes the one proposed in [15] for IMCs, allows choosing a different correspondence matrix for each probability distribution satisfying the constraint:

Definition 7 (Weak Refinement)**.** *Let $S_1 = \langle\{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle\{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be CMCs with $A_2 \subseteq A_1$. The relation $\mathcal{R} \subseteq \{1,\dots,k_1\} \times \{1,\dots,k_2\}$ is a* weak refinement relation *iff whenever $v\,\mathcal{R}\,u$ then:*

1) *$V_1(v)\downarrow_{A_2} \subseteq V_2(u)$ and*
2) *for any distribution $x \in [0,1]^{1\times k_1}$ satisfying $\varphi_1(v)(x)$, there exists a matrix $\Delta \in [0,1]^{k_1\times k_2}$ such that*
   - *For all $1 \le i \le k_1$, $x_i \neq 0 \implies \sum_{j=1}^{k_2} \Delta_{ij} = 1$;*
   - *$\varphi_2(u)(x \times \Delta)$ holds and*
   - *if $\Delta_{v'u'} \neq 0$ then $v'\,\mathcal{R}\,u'$.*

CMC S_1 (weakly) refines S_2, written $S_1 \preceq S_2$, iff $o_1\,\mathcal{R}\,o_2$.

Example. *Figure 3d illustrates a family of correspondence matrices parametrized by γ, witnessing the weak refinement between initial states of S_3 and S_4 (defined in Figures 3a–3b). The actual matrix used in proving the weak refinement depends on the probability distribution vector z that satisfies the constraint function φ_3 of state $(1,1)$. Take $\gamma = \frac{0.7-z_{22}}{z_{23}}$ if $z_{22} \le 0.7$ and $\gamma = \frac{0.8-z_{22}}{z_{23}}$ otherwise ($z_{22} \le 0.8$ by definition). It is easy to see that $\varphi_3((1,1))(z)$ implies $\varphi_4(1)(z\times\Delta)$.*

Both weak and strong refinements imply implementation set inclusion (see [16]). In Section V, we shall see that the converse holds for a particular class of CMCs. However, this is not the case in general: strong refinement is strictly stronger than weak refinement, which is strictly stronger than implementation set inclusion. Formally, we have the following proposition.

Proposition 8. *There exist CMCs S_a, S_b, S_c and S_d such that*

- *S_a weakly refines S_b, and S_a does not strongly refine S_b;*
- *$[\![S_c]\!] \subseteq [\![S_d]\!]$, and S_c does not weakly refine S_d.*

So our refinement relations for CMCs can be ordered from finest to coarsest: the strong refinement, the weak refinement, and the implementation set inclusion. As the implementation set inclusion is the *ultimate* refinement, checking finer refinements is used as a pragmatic syntax-driven, but sound, way of deciding it. Algorithms for checking strong and weak refinements are discussed in [16]. Those algorithms are polynomial in the number of state, but the treatment of each state depends on the complexity of the constraints. Finally, let us mention that lower-bounds for the strong and weak refinement checking remain open problems.

Conjunction. *Conjunction*, also called *logical composition*, combines requirements of several specifications.

Definition 9 (Conjunction)**.** *Let $S_1 = \langle\{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle\{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be two CMCs. The conjunction of S_1 and S_2, written $S_1 \wedge S_2$, is the CMC $S = \langle\{1,\dots,k_1\} \times \{1,\dots,k_2\}, (o_1, o_2), \varphi, A, V\rangle$ with $A = A_1 \cup A_2$, $V((u,v)) = V_1(u)\uparrow^A \cap V_2(v)\uparrow^A$, and*

$$\begin{aligned}\varphi((u,v))(x_{1,1}, x_{1,2}, \dots, x_{2,1}, \dots, x_{k_1,k_2}) \equiv\\ \varphi_1(u)(\textstyle\sum_{j=1}^{k_2} x_{1,j}, \dots, \sum_{j=1}^{k_2} x_{k_1,j}) \wedge\\ \varphi_2(v)(\textstyle\sum_{i=1}^{k_1} x_{i,1}, \dots, \sum_{i=1}^{k_1} x_{i,k_2}).\end{aligned}$$

Conjunction may introduce inconsistent states and thus its use should normally be followed by applying the pruning operator β^*. As already stated in the introduction, the result of conjoining two IMCs is not an IMC in general, but a CMC whose constraint functions are systems of linear inequalities. Figure 3b depicts a CMC S_3 expressing the conjunction of IMCs S_1 and S_2 (see Figures 2a–2b). The constraint $z_{2,3} + z_{3,3} \ge 0.2$ in state $(1,1)$ cannot be expressed as an interval.

As expected, conjunction of two specifications coincides with their greatest lower bound with respect to the weak refinement (also called *shared refinement*).

Theorem 10. *Let S_1, S_2 and S_3 be three CMCs. We have (a) $((S_1 \wedge S_2) \preceq S_1)$ and $((S_1 \wedge S_2) \preceq S_2)$ and (b) if $(S_3 \preceq S_1)$ and $(S_3 \preceq S_2)$, then $S_3 \preceq (S_1 \wedge S_2)$.*

In fact, as follows from the later results of Section V, the set of implementations of a conjunction of two *deterministic* specifications S_1 and S_2 coincides with the intersection of implementation sets of S_1 and S_2 (the greatest lower bound in the lattice of implementation sets).

IV. Compositional Reasoning using the Principle of Separation of Concerns

Let us now turn to *structural* composition. In our theory, as we already said in the introduction and after presenting CMCs, choices regarding the set of valuations and stochastic choices are independent from each others. This property of the model naturally leads to a definition of the parallel composition operator based on the principle of *separation of concerns*. The idea is that probabilistic behaviours are composed separately from the synchronization of the sets of state valuations. This allows realizing probabilistic composition as a simple product of independent distributions.

Remark 1. *The principle of separation of concerns is intensively used in the definition of parallel composition for many systems that mix stochastic and non-deterministic choices. Among them, one can cite many theories for probabilistic process algebra [11], [13]. Similar principles are also applied for continuous time stochastic models, in a slightly different setting based on CTMCs [14]. In Section VII, we shall see that our structural composition covers the one of probabilistic automata.*

Following the separation of concerns principle, components are composed first into a product (or effectively just a vector of independent entities), and then synchronized by constraining their behaviour. This design is both simple and expressive: it allows applying diverse synchronization mechanisms, beyond just matching inputs to outputs. Moreover it elegantly exploits the prior knowledge on logical composition, as the synchronization operator turns out to be realizable using conjunction.

We start by discussing how systems and specifications can be composed in a non-synchronizing way, then we introduce a notion of synchronization. The non-synchronizing *independent* composition is largely just a product of two MCs (or CMCs).

Definition 11 (Parallel Composition of MCs)**.** *Let $P_1 = \langle \{1,\dots,n_1\}, o_1, M', A_1, V_1\rangle$ and $P_2 = \langle \{1,\dots,n_2\}, o_2, M'', A_2, V_2\rangle$ be two MCs with $A_1 \cap A_2 = \emptyset$. The parallel composition of P_1 and P_2 is the MC $P_1 \parallel P_2 = \langle \{1,\dots,n_1\} \times \{1,\dots,n_2\}, (o_1,o_2), M, A_1 \cup A_2, V\rangle$ where: $M \in [0,1]^{(n_1\times n_2)\times(n_1\times n_2)}$ is such that $M_{(p,q)(r,s)} = M'_{pr} \cdot M''_{qs}$; and $V((p,q)) = V_1(p) \cup V_2(q)$.*

For CMCs, we have the following definition.

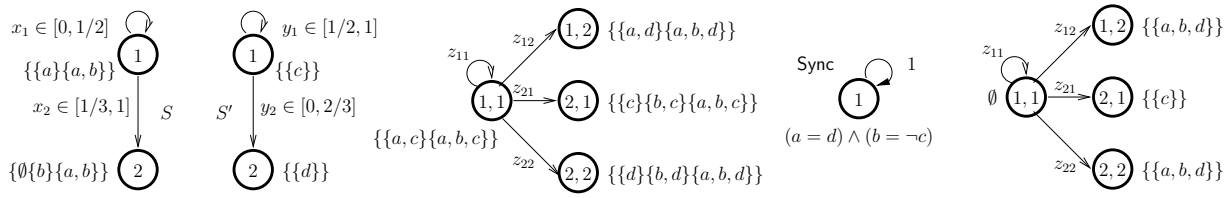
Definition 12 (Parallel Composition of CMCs)**.** *Let $S_1 = \langle \{1,\dots,k_1\}, o_1, \varphi_1, A_1, V_1\rangle$ and $S_2 = \langle \{1,\dots,k_2\}, o_2, \varphi_2, A_2, V_2\rangle$ be CMCs with $A_1 \cap A_2 = \emptyset$. The parallel composition of S_1 and S_2 is the CMC $S_1 \parallel S_2 = \langle \{1,\dots,k_1\} \times \{1,\dots,k_2\}, (o_1,o_2), \varphi, A_1 \cup A_2, V\rangle$, where $\varphi((u,v))(z_{1,1}, z_{1,2}, \dots z_{2,1}, \dots, z_{k_1,k_2}) = \exists x_1,\dots,x_{k_1}, y_1,\dots,y_{k_2} \in [0,1]$ such that $\forall (i,j) \in \{1,\dots,k_1\} \times \{1,\dots,k_2\}$ we have $z_{i,j} = x_i \cdot y_j$ and $\varphi_1(u)(x_1,\dots,x_{k_1}) = \varphi_2(v)(y_1,\dots,y_{k_2}) = 1$. Finally, $V((u,v)) = \{Q_1 \cup Q_2 \mid Q_1 \in V_1(u), Q_2 \in V_2(v)\}$.*

It is worth mentioning that IMCs are not closed under composition. Consider IMCs S and S' given in Figure 4a and their composition $S \parallel S'$ given in Figure 4b. Assume first that $S \parallel S'$ is an IMC. As a variable z_{ij} is the product of two variables x_i and y_j, if $S \parallel S'$ is an IMC, then one can show that the interval for z_{ij} is obtained by computing the products of the bounds of the intervals over which x_i and y_j range. Hence, we can show that $z_{11} \in [0,1/2], z_{12} \in [0,1/3], z_{21} \in [1/6,1], z_{22} \in [0,2/3]$. Let $[a,b]$ be the interval for the constraint z_{ij}, it is easy to see that there exist implementations I_1 of S_1 and I_2 of S_2 such that $I_1 \parallel I_2$ satisfies the constraint $z_{ij} = a$ (resp. $z_{ij} = b$). However, while each bound of each interval can be satisfied independently, some points in the polytope defined by the intervals and the constraint $\sum z_{ij} = 1$ cannot be reached. As an example, consider $z_{11} = 0, z_{12} = 1/3, z_{21} = 1/3, z_{22} = 1/3$. It is clearly inside the polytope, but one cannot find an implementation I of $S \parallel S'$ satisfying the constraints given by the parallel composition. Indeed, having $z_{11} = 0$ implies that $x_1 = 0$ and thus that $z_{12} = 0$.

Theorem 13. *If S'_1, S'_2, S_1, S_2 are CMCs then $S'_1 \preceq S_1$ and $S'_2 \preceq S_2$ implies $S'_1 \parallel S'_2 \preceq S_1 \parallel S_2$, so the weak refinement is a precongruence with respect to parallel composition. Consequently, for any MCs P_1 and P_2 we have that $P_1 \models S_1 \wedge P_2 \models S_2$ implies $P_1 \| P_2 \models S_1 \| S_2$.*

As alphabets of composed CMCs have to be disjoint, the composition does not synchronize the components on state valuations like it is typically done for other (non-probabilistic) models. However, synchronization can be introduced by conjoining the composition with a *synchronizer*—a single-state CMC whose valuation function relates the atomic propositions of the composed CMCs.

Example. *CMC $S \parallel S'$ of Figure 4b is synchronized with the synchronizer* Sync *given in Figure 4c.* Sync *removes from $S \parallel S'$ all the valuations that do not satisfy $(a = d) \wedge (b = \neg c)$. The result is given in Figure 4d. Observe that an inconsistency appears in State $(1,1)$. Indeed, there is no implementation of the two CMCs that can synchronize in the prescribed way. In general inconsistencies like this one can be uncovered by applying the pruning operator, which would return an empty specification. So synchronizers enable discovery of incompatibilities between component specifications in the same way as it is known for non-probabilistic specification models.*



(a) Two CMCs S and S' (b) $S \parallel S'$ (c) Synchronizer Sync (d) $(S \parallel S') \wedge$ Sync

Fig. 4: Parallel composition and synchronization of CMCs

Synchronization is associative with respect to composition, which means that the order of synchronization and composition is inessential for final functionality of the system.

Theorem 14. *Let S_1, S_2 and S_3 be three CMCs with pairwise disjoint sets of propositions A_1, A_2 and A_3. Let Sync_{123} be a synchronizer over $A_1 \cup A_2 \cup A_3$ and let Sync_{12} be the same synchronizer with its set of propositions restricted to $A_1 \cup A_2$. The following holds $[\![[((S_1 \parallel S_2) \wedge \mathsf{Sync}_{12}) \parallel S_3] \wedge \mathsf{Sync}_{123}]\!] = [\![(S_1 \parallel S_2 \parallel S_3) \wedge \mathsf{Sync}_{123}]\!]$.*

Finally, synchronized composition also supports component-based refinement in the style of Theorem 13:

Theorem 15. *If S_1', S_2', S_1, S_2 are CMCs,* Sync *is a synchronizer and $S_1' \preceq S_1 \wedge S_2' \preceq S_2$ then $(S_1' \parallel S_2') \wedge \mathsf{Sync} \preceq (S_1 \parallel S_2) \wedge \mathsf{Sync}$.*

Consequently, a modeller can continue independent refinement of specifications under synchronization, knowing that the original synchronized specification will not be violated.

V. Deterministic CMCs

Clearly, if all implementations of a specification S_1 also implement a specification S_2, then the former is a proper strengthening of the latter. Indeed, S_1 specifies implementations that break no assumptions that can be made about implementations of S_2. Thus implementation set inclusion *is* a desirable refinement for specifications. Unfortunately, this problem is still open, and, as we have said, the weak and the strong refinement soundly approximate it. Had that approximation been complete, we would have had an effective decision procedure for implementation set inclusion. In this section, we argue that this indeed is the case for an important subclass of specifications: *deterministic CMCs*. A CMC S is *deterministic* iff for every state i, states reachable from i have pairwise disjoint admissible valuations:

Definition 16. *Let $S = \langle \{1, \ldots, k\}, o, \varphi, A, V \rangle$ be a CMC. S is* deterministic *iff for all states $i, u, v \in \{1, \ldots, k\}$, if there exists $x \in [0,1]^k$ such that $(\varphi(i)(x) \wedge (x_u \neq 0))$ and $y \in [0,1]^k$ such that $(\varphi(i)(y) \wedge (y_v \neq 0))$, then we have that $V(u) \cap V(v) = \emptyset$.*

In Figures 2a and 2b, both S_1 and S_2 are deterministic specifications. In particular states 2 and 3, reachable from 1 in both CMCs, have disjoint constraints on valuations. On the other hand, the CMC T given in Figure 5 is non-deterministic. Indeed, for States 2 and 3, which can both be reached from State 1, we have that $V_T(2) \cap V_T(3) = \{\{a, c\}\} \neq \emptyset$.

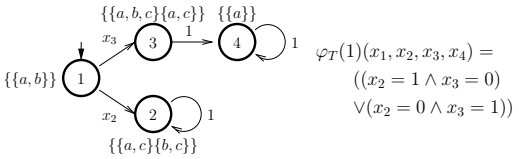


Fig. 5: A CMC T whose set of implementations cannot be represented with a deterministic CMC

Deterministic CMCs are less expressive than non-deterministic ones, in the sense that the same implementation sets cannot sometimes be expressed. Consider again the CMC T given in Figure 5. It is such that its set of implementations cannot be represented by a deterministic CMC. Indeed, any merging of States 2 and 3 in T would result in a CMC that accepts models where one can loop on valuation $\{a, c\}$ and then accept valuation $\{a\}$ with probability 1. Such a model cannot be accepted by T.

Proposition 17. *Conjunction and composition preserve determinism.*

In [16], we present a determinization algorithm that can be applied to any CMC S whose initial state is a single valuation set. The result of the algorithm is a new CMC weakly refined by S. Consequently the implementation set of the result includes the one of S (see [16]). This character of determinization resembles the known determinization algorithms for modal transition systems [19].

We now state one of the main theorems of the paper: the weak refinement is complete with respect to implementation set inclusion for deterministic CMCs:

Theorem 18. *Let $S_1 = \langle \{1, \ldots, k_1\}, o_1, \varphi_1, A_1, V_1 \rangle$ and $S_2 = \langle \{1, \ldots, k_2\}, o_2, \varphi_2, A_2, V_2 \rangle$ be two consistent single valuation normal form deterministic CMCs with $A_2 \subseteq A_1$. We have $[\![S_1]\!] \subseteq [\![S_2]\!] \Rightarrow S_1 \preceq S_2$.*

Proof: We present a sketch of the proof and refer to [16] for details. We construct the refinement relation by relating all pairs of states of S_1 and S_2 for which implementation inclusion holds. Let $\mathcal{R} \subseteq \{1, \ldots, k_1\} \times \{1, \ldots, k_2\}$ such that $v \mathcal{R} u$ iff for all MC I and state p of I we have $p \models v \Rightarrow p \models u$.

As we consider pruned CMCs, there exist implementations for all states. Then the usual, albeit complex and long in this case, coinductive proof technique is applied, showing that this relation is indeed a weak refinement relation. The crucial point of the argument lies in proving the closure property — i.e. that if a S_1 state u advances possibly to u', then indeed the corresponding state v of S_2 can also advance to v' and the (u', v') pair is in $\mathcal{R}$. In other words that implementation inclusion of predecessors implies the implementation inclusion of successors. This is proven in an ad absurdum argument. Roughly, assume that there would exist an implementation I' of u' which is not an implementation of v'. Then one can construct an implementation I'' of u which evolves as I'. This implementation would not implement v' but could implement some other state of S_2. This case is ruled out by requiring determinism and a normal form of S_2. Then the only way for I'' to evolve is to satisfy v' which contradicts the assumption that I' is not an implementation of v'. ∎

Since any consistent CMC with a single valuation in initial state can be normalized, Theorem 18 holds even if S_1 and S_2 are not in single valuation normal form. Thus, weak refinement and the implementation set inclusion coincide on the class of deterministic CMCs with at most single valuation in the initial state. Finally, Theorem 18 also holds for strong refinement. Indeed, the following theorem states that weak and strong refinements coincide on the class of deterministic CMCs.

Theorem 19. *Let $S_1 = \langle\{1, \ldots, k_1\}, o_1, \varphi_1, A, V_1\rangle$ and $S_2 = \langle\{1, \ldots, k_2\}, o_2, \varphi_2, A, V_2\rangle$ be two deterministic CMCs in normal form. If there exists a weak refinement relation $\mathcal{R}$ such that $S_1 \mathcal{R} S_2$, then $\mathcal{R}$ is also a strong refinement relation.*

Finally, the above results on completeness for deterministic specifications carry over to refinements of [9] and [15], which are special cases of our refinements. Completeness properties for these refinements were open problems until now.

Discussion: A weaker Definition of Determinism. Our notion of determinism may look too strong. Indeed, it assumes that, from a given state i, one cannot reach two states u and v that share common sets of valuations. The assumption is made independently of the distributions used to reach the two states, i.e., it may be the case that there exists no distribution in where both u and v can be reached simultaneously. A natural way to solve the problem would be to consider a weaker version of determinism. More precisely, we say that a CMC $S = \langle\{1, \ldots, k\}, o, \varphi, A, V\}$ is weakly deterministic if whenever there exists $x \in [0, 1]^k$ and states i, u, v such that $\varphi(i)(x)$ and $x_u > 0$ and $x_v > 0$, we have $V(u) \cap V(v) = \emptyset$. This version of determinism is weaker than the one given in Definition 16. Indeed, only states that can be reached by the same distribution should have disjoint sets of valuations. Though this notion seems reasonable, one can show (see [16] for a proof) that there exist two weakly deterministic CMCs S_c and S_d such that S_c thoroughly but not weakly refines S_d. Hence working with this weaker, but natural, version of determinism does not close the gap between weak and thorough refinements.

VI. Polynomial CMCs

It is not surprising that CMCs are closed under both logical and structural compositions. Indeed, CMCs do not make any assumptions on constraint functions. There are however many classes of constraints that are practically intractable. While this paper is mainly concerned with the development of the theoretical foundations for CMCs, we now briefly study classes of CMCs for which operations on constraints required by our algorithms can be managed quite efficiently.

A first candidate could be linear constraints, which is the obvious generalization of interval constraints. Unfortunately, linear constraints CMCs are not closed under structural composition. Indeed, as we have seen in Section IV the composition of two linear constraints leads to a polynomial constraint. However, what is more interesting is that polynomial constraints *are* closed under both logical and structural composition and that these operations do not increase the quantifier alternations since they only introduce existential quantifiers. Hence, one can claim that CMCs with polynomial constraints and only existential quantifiers are certainly the smallest extension of IMCs closed under all operations.

From the algorithmic point of view, working with polynomial constraints should not be seen as an obstacle. First, we observe that algorithms for logical and structural composition do not require any complex operations on polynomials. The refinements algorithms (presented in [16]) are polynomial in the number of states, and each iteration requires a quantifier elimination. This procedure is known to be double exponential in general, but there exist efficient single exponential algorithms [20], [21] when quantifier alternations are fixed. Those algorithms are implemented in Maple [22]. The pruning operation is polynomial in the number of states, but each iteration also requires an exponential treatment as one has to decide whether the constraints have at least a solution. Again, such problem can be solved with efficient algorithms. Finally, determinizing a CMC can be performed with a procedure that is similar to the determinization procedure for finite-state automata. Such a procedure is naturally exponential in the number of states.

VII. Discussion of Refinement and Composition

CMCs are a newcomer in a long series of probabilistic modeling languages and abstractions for them. Throughout the paper we have indicated that many of our results directly translate to simpler abstractions, like IMCs. We shall now further discuss this foundational aspect of CMCs, showing how they subsume a few established notions of refinement and composition for probabilistic automata (and for process algebra based on them).

Below we write $\mathrm{Dist}(S)$ for the set of all probability distributions over a finite set S. Given two sets S and T and a probability distribution $\alpha \in \mathrm{Dist}(S \times T)$, we denote the marginal distribution over S as $\alpha_{s,T} = \sum_{t \in T} \alpha_{s,t}$, and similarly for T. We say that φ is a *non-deterministic*

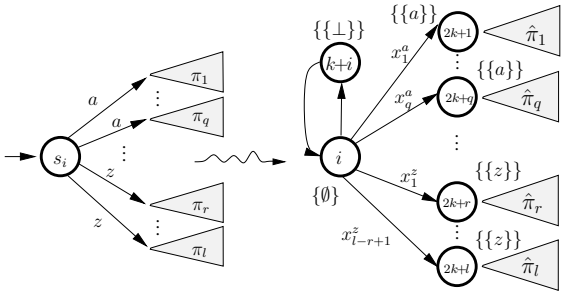


Fig. 6: Reducing a PA to CMC. There $\widehat{\pi}$ denotes a distribution constraint, which has a unique solution π.

distribution constraint over set I if all solutions x of φ are point distributions; so $\exists i.\, x_i = 1$. Write $[\,{}^{i}_{S}\,]$ to denote a particular point distribution for which $[\,{}^{i}_{S}\,]_i = 1$. Notice that non-deterministic distribution constraints model a non-deterministic choice of an element from S. They will be used to encode non-determinism in CMCs.

A probabilistic automaton (PA for short) [11] is a tuple $\mathbb{S} = (S, Act, \rightarrow, s_1)$, where S is a finite set of states, $\rightarrow \subseteq S \times Act \times \mathrm{Dist}(S)$ is a finite transition relation and $s_1 \in S$ is the initial state. The *derived combined transition relation* of $\mathbb{S}$ is given by $\rightarrow_c \in S \times \mathrm{Act} \times \mathrm{Dist}(S)$. If $\pi \in \mathrm{Dist}(S)$ and $\varrho \in \mathrm{Dist}(T)$ then $\pi \otimes \varrho$ denotes the unique independent product distribution such that $(\pi \otimes \varrho)_{s,t} = \pi_s \cdot \varrho_t$.

We say that $t \xrightarrow{a}_c \varrho$ iff ϱ is a convex linear combination of vectors from $\boldsymbol{\varrho} = \{\varrho_i \mid t \xrightarrow{a} \varrho_i\}$, so $\varrho = \boldsymbol{\varrho} \times \lambda$, where λ is a distribution vector $\lambda \in [0,1]^{|\boldsymbol{\varrho}|}$. We interpret $\boldsymbol{\varrho}$ as a matrix, where ith column is a distribution ϱ_i.

Consider two PA $\mathbb{S} = (S, Act, \rightarrow^S, s_0)$ and $\mathbb{T} = (T, Act, \rightarrow^T, t_0)$. For a binary relation $R \subseteq S \times T$ we define a derived relation $R^* \subseteq \mathrm{Dist}(S) \times \mathrm{Dist}(T)$ such that $\pi R^* \varrho$ iff there exists a distribution $\alpha \in \mathrm{Dist}(S \times T)$ and (1) $\alpha_{q,T} = \pi_q$ for all $q \in S$, (2) $\alpha_{S,r} = \varrho_r$ for all $r \in T$ and (3) $\alpha_{s,t} \neq 0$ implies sRt.

Definition 20 (Simulation [11])**.** *A relation $R \subseteq S \times T$ is a* simulation *iff $(s,t) \in R$ implies that whenever $s \xrightarrow{a} \pi$ for a distribution π, then $t \xrightarrow{a} \varrho$ for distribution ϱ such that $\pi R^* \varrho$.*

R is a probabilistic *simulation iff $(s,t) \in R$ implies that if $s \xrightarrow{a} \pi$ then $t \xrightarrow{a}_c \varrho$ for some distribution ϱ, and $\pi R^* \varrho$.*

Let $A \subseteq Act$ be the subset of actions on which $\mathbb{S}$ and $\mathbb{T}$ should synchronize. The *parallel composition* of $\mathbb{S}$ and $\mathbb{T}$ is a PA $\mathbb{S} \parallel \mathbb{T} = (S \times T, Act, \rightarrow, (s_0, t_0))$, where $\rightarrow$ is the largest transition relation such that $(s,t) \xrightarrow{a} \pi \otimes \varrho$ if:

$a \in A$ and $s \xrightarrow{a}{}^S \pi$ and $t \xrightarrow{a}{}^T \varrho$, or
$a \notin A$ and $s \xrightarrow{a}{}^S \pi$ and $\varrho = [\,{}^{t}_{T}\,]$, or
$a \notin A$ and $\pi = [\,{}^{s}_{S}\,]$ and $t \xrightarrow{a}{}^T \varrho$.

We now propose a linear encoding of PAs into CMCs, which reduces simulation and composition of PAs to refinement and composition of CMCs (see Fig. 6). Let $\mathbb{S} = (\{s_1, \ldots, s_k\}, Act, \rightarrow, s_0)$ be a PA. And let l be the number of reachable action-distribution pairs, so $\Omega_{\mathbb{S}} = \{(a_1, \pi_1), \ldots, (a_l, \pi_l)\} = \{(a, \pi) \mid \exists s \in S.\, s \xrightarrow{a} \pi\}$. The corresponding CMC is $\widehat{\mathbb{S}} = (\{1, \ldots, 2k{+}l\}, 1, \widehat{\varphi}, \mathrm{Act} \cup \bot, \widehat{V})$, where $\bot \notin Act$. $\widehat{\mathbb{S}}$ has three kinds of states. Type-1 states, $1 \ldots k$, correspond directly to states of $\mathbb{S}$. Distributions leaving these states model a non-deterministic choice. Type-2 states, $k+1, \ldots, 2k$, model a possibility that a component remains idle in a state. Type-3 states, $2k{+}1, \ldots, 2k{+}l$ model the actual distributions of $\mathbb{S}$.

$\widehat{V}$ assigns value $\{\emptyset\}$ to type-1 states and value $\{\{\bot\}\}$ to type-2 states. For type-3: $\widehat{V}(2k{+}i') = \{\{a_{i'}\}\}$ for $1 \leq i' \leq l$. The distribution constraints are as follows:

$\widehat{\varphi}(i)(x)$ if i is type-1 and
$\quad x = [\,{}^{k+i}_{1..2k+l}\,]$ or $s_i \xrightarrow{a_{i'}} \pi_{i'} \wedge x = [\,{}^{2k+i'}_{1..2k+l}\,]$ for $1 \leq i' \leq l$.
$\widehat{\varphi}(k+i)(x)$ if $k{+}i$ is type-2 and $x = [\,{}^{i}_{1..2k+l}\,]$.
$\widehat{\varphi}(2k+i')(x)$ if $2k+i'$ is type-3 and $x = \pi_{i'}$

We can now relate simulation of PA to refinement of CMCs:

Theorem 21. *$\mathbb{T}$ simulates $\mathbb{S}$ iff $\widehat{\mathbb{S}}$ strongly refines $\widehat{\mathbb{T}}$.*

Another, very similar, but slightly more complicated, encoding exists, for which weak refinement coincides with *probabilistic* simulation. See [16] for details.

The same encoding is used to characterize parallel composition of PAs using parallel composition of CMCs.

Theorem 22. *For two PAs $\mathbb{S}$ and $\mathbb{T}$ over the same set of actions Act and a set of synchronizing actions $A \subseteq Act$ we have that $\widehat{\mathbb{S} \parallel \mathbb{T}}$ is isomorphic to*

$$((\widehat{\mathbb{S}} \parallel \widehat{\mathbb{T}}[{}^{a'}/_{a}]_{a \in Act}) \wedge \mathsf{S}_A)\,[{}^{a}/_{(a,a')};\ {}^{a}/_{(a,\bot')};\ {}^{a}/_{(\bot,a')}]_{a \in Act}$$

where S_A is a synchronizer over $Act_\bot \times Act'_{\bot'}$ defined by

$$(\forall a \in A.\, a \iff a') \wedge (\forall a \notin A.\, (a \implies \bot') \wedge (a' \implies \bot))$$

Expression $S[{}^{a'_1}/_{a_1}; \ldots; {}^{a'_n}/_{a_n}]_{a_1,\ldots,a_n \in Act}$ denotes a substitution, substituting a primed version of name a_i for each occurrence in a_i, for all actions in *Act*.

Interestingly, the precongruence property for the parallel composition of PAs is obtained for free as a corollary of the above two reduction theorems and Thm. 13. Similarly, we obtain precongruence with probabilistic simulation using a suitable encoding—a good example how CMCs can be used to study properties of simpler languages in a generic way.

VIII. Related Work and Concluding Remarks

We have presented CMCs—a new model for representing a possibly infinite family of MCs. Unlike the previous attempts [9], [15], our model is closed under many design operations, including composition and conjunction. We have studied these operations as well as several classical compositional reasoning properties, showing that, among others, the CMC specification theory is equipped with a complete refinement relation (for deterministic specifications), which naturally interacts with parallel composition, synchronization and conjunction. We have also demonstrated how our framework can be used to obtain properties for less expressive languages, by using reductions.

Two recent contributions [15], [23] are related to our work. Fecher et al. [15] propose a model checking procedure for PCTL [24] and Interval Markov Chains (other procedures recently appear in [18], [25]), which is based on weak refinement. However, our objective is not to use CMCs within a model checking procedure for probabilistic systems, but rather as a specification theory.

Very recently Katoen and coauthors [23] have extended Fecher's work to *Interactive* Markov Chains, a model for performance evaluation [26], [27]. Their abstraction uses the continuous time version of IMCs [28] augmented with may and must transitions, very much in the spirit of [3]. Parallel composition is defined and studied for this abstraction, however conjunction has been studied neither in [15] nor in [23].

Over the years process algebraic frameworks have been proposed for describing and analyzing probabilistic systems based on Markov Chains (MCs) and Markov Decision Processes [14], [29], [30]. Also a variety of probabilistic logics have been developed for expressing properties of such systems, e.g., PCTL [10]. Both traditions support refinement between specifications using various notions of probabilistic simulation [15], [9] and, respectively, logical entailment [31]. Whereas the process algebraic approach favors structural composition (parallel composition), the logical approach favors logical composition (conjunction). Neither of the two supports *both* structural and logical composition.

As a future work, it would be of interest to design, implement and evaluate efficient algorithms for procedures outlined in this paper. We will also study the decidability of the set inclusion problem. We would also like to define a quotient relation for CMCs, presumably building on results presented in [32]. The quotienting operation is of particular importance for component reuse. One could also investigate applicability of our approach in model checking procedures, in the same style as Fecher and coauthors have used IMCs for model checking PCTL [15]. Finally, it would be interesting to extend our composition operation by considering products of dependent probability distributions in the spirit of [33].

Acknowledgements

This work was supported by the European STREP-COMBEST project no. 215543, by VKR Centre of Excellence MT-LAB, and by an "Action de Recherche Collaborative" ARC (TP)I.

References

[1] T. A. Henzinger and J. Sifakis, "The embedded systems design challenge," in *FM*, ser. lncs, vol. 4085. Springer, 2006, pp. 1–15.

[2] L. de Alfaro and T. A. Henzinger, "Interface-based design," in *Engineering Theories of Software-intensive Systems*, ser. NATO Science Series: Mathematics, Physics, and Chemistry, vol. 195. Springer, 2005, pp. 83–104.

[3] K. G. Larsen, "Modal specifications," in *AVMS*, ser. LNCS, vol. 407. Springer, 1989, pp. 232–246.

[4] J.-B. Raclet, E. Badouel, A. Benveniste, B. Caillaud, A. Legay, and R. Passerone, "Modal interfaces: Unifying interface automata and modal cifications," in *EMSOFT*, 2009.

[5] K. G. Larsen, U. Nyman, and A. Wasowski, "Modal I/O automata for interface and product line theories," in *ESOP*, ser. LNCS. Springer, 2007, pp. 64–79.

[6] L. de Alfaro and T. A. Henzinger, "Interface automata," in *FSE*. ACM Press, 2001, pp. 109–120.

[7] L. Doyen, T. A. Henzinger, B. Jobstmann, and T. Petrov, "Interface theories with component reuse," in *EMSOFT*. ACM Press, 2008, pp. 79–88.

[8] A. Chakrabarti, L. de Alfaro, T. A. Henzinger, and F. Y. C. Mang, "Synchronous and bidirectional component interfaces," in *CAV*, ser. LNCS, vol. 2404. springer, 2002, pp. 414–427.

[9] B. Jonsson and K. G. Larsen, "Specification and refinement of probabilistic processes," in *LICS*. IEEE Computer, 1991.

[10] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Asp. Comput.*, vol. 6, no. 5, 1994.

[11] R. Segala and N. Lynch, "Probabilistic simulations for probabilistic processes," in *CONCUR*, ser. LNCS, vol. 836. springer, 1994, pp. 481–496.

[12] H. Hansson and B. Jonsson, "A calculus for communicating systems with time and probabitilies," in *IEEE Real-Time Systems Symposium*, 1990, pp. 278–287.

[13] B. Jonsson, K. Larsen, and W. Yi, "Probabilistic extensions of process algebras," in *Handbook of Process Algebra*. Elsevier, 2001, pp. 681–710.

[14] H. Hermans, *Interactive Markov Chains*, verlag, Ed. Springer, 2002.

[15] H. Fecher, M. Leucker, and V. Wolf, "Don't Know in probabilistic systems," in *SPIN*, ser. LNCS, vol. 3925, 2006.

[16] B. Caillaud, B. Delahaye, K. G. Larsen, A. Legay, M. L. Pedersen, and A. Wasowski, "Compositional Design Methodology with Constraint Markov Chains," INRIA, Research Report RR-6993, 2009. [Online]. Available: http://hal.inria.fr/inria-00404304/en/

[17] K. Sen, M. Viswanathan, and G. Agha, "Model-checking Markov chains in the presence of uncertainties," in *TACAS*, ser. LNCS, vol. 3920. Springer, 2006, pp. 394–410.

[18] K. Chatterjee, K. Sen, and T. A. Henzinger, "Model-checking omega-regular properties of interval Markov chains," in *FoSSaCS*, ser. LNCS, vol. 4962. Springer, 2008.

[19] N. Benes, J. Kretinsky, K. G. Larsen, and J. Srba, "On determinism in modal transition systems," to appear in TCS.

[20] C. W. Brown, "Simple CAD construction and its applications," *Journal of Symbolic Computation*, vol. 31, no. 5, 2001.

[21] C. W. Brown and J. H. Davenport, "The complexity of quantifier elimination and cylindrical algeraic decomposition," in *SSAC*, Waterloo, ON, Canada, 2007, pp. 54–60.

[22] H. Yanami and H. Anai, "SyNRAC: a Maple toolbox for solving real algebraic constraints," *ACM Communications in Computer Algebra*, vol. 41, no. 3, pp. 112–113, September 2007.

[23] J. Katoen, D. Klink, and M. R. Neuhäußer, "Compositional abstraction for stochastic systems," in *FORMATS*, ser. LNCS, vol. 5813. Springer, 2009, pp. 195–211.

[24] F. Ciesinski and M. Größer, "On probabilistic computation tree logic," in *VSS*, ser. LNCS, vol. 2925. Springer, 2004.

[25] S. Haddad and N. Pekergin, "Using stochastic comparison for efficient model checking of uncertain Markov chains," in *QEST*. IEEE Computer Society Press, 2009, pp. 177–186.

[26] H. Hermanns, U. Herzog, and J. Katoen, "Process algebra for performance evaluation," *Theor. Comput. Sci.*, vol. 274, no. 1-2, pp. 43–87, 2002.

[27] J. Hillston, *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[28] J. Katoen, D. Klink, M. Leucker, and V. Wolf, "Three-valued abstraction for continuous-time Markov chains," in *CAV*, ser. LNCS, vol. 4590. Springer, 2007, pp. 311–324.

[29] S. Andova, "Process algebra with probabilistic choice," in *ARTS*, ser. LNCS, vol. 1601. Springer, 1999.

[30] N. López and M. Núñez, "An overview of probabilistic process algebras and their equivalences," in *VSS*, ser. LNCS, vol. 2925. Springer, 2004, pp. 89–123.

[31] H. Hermanns, B. Wachter, and L. Zhang, "Probabilistic CEGAR," in *CAV*, ser. LNCS, vol. 5123. Springer, 2008.

[32] K. G. Larsen and A. Skou, "Compositional verification of probabilistic processes," in *CONCUR*, ser. LNCS, vol. 630. Springer, 1992, pp. 456–471.

[33] L. de Alfaro, T. A. Henzinger, and R. Jhala, "Compositional methods for probabilistic systems," in *CONCUR*, ser. lncs, vol. 2154. Springer, 2001, pp. 351–365.